Fahrenheit Co.

Communities

Final Project Report

Eric Anderson - Implementation, Functionalities & References
Christopher Bednarz - UML Diagrams
Zoe Chamlee - Performance Metrics
James W. Garrett - Editing, Executive Sumamry, Conclusion
Mason Holley - Code Blocks
Rishabh Tewari - Gantt Chart & Future Plans

Date of Submission: November 22th, 2021

Executive Summary

Communities is an app that allows people within the local scene to have an easier time interacting with each other. Our program does this by isolating the scope of the communities to the users' local areas. This app consists of a map of the user's nearby surroundings that will allow them to browse and join local groups and communities. There is also a feed that will post all info related to the communities that the user joined.

Project Goals and Objectives

Our focus in creating Communities is to provide a platform for local communities to grow and flourish in a healthy environment. Communities will allow users to connect with or create a local scene for all kinds of interests, hobbies, or activities. With each user that joins our platform, we gain a new potential member for one of the many communities. While a user might join for one specific community, we want them to be able to find many communities which might fit their tastes all in their local area. With this, we believe our company, Fahrenheit Co. will be able to cultivate a healthy user-base which will allow continuous app growth.

We have diversified from other similar applications such as Discord and Reddit by location-locking our communities. Each community will have the tools available to announce and schedule events related to each community which will be easily accessible and known to all members. We will also be working to provide tools to allow the creators or appointed leaders of a community to maintain and moderate the community.

Proposed Product

Our proposed product is an app that allows the user to view communities that are geographically close to his current location. Through our app, users will have the ability to browse the map of their city(or any nearby city) and select any communities that appeal to them. Once the communities are selected, the user will be able to see all of the posts, news, and events that are related to the communities.

Project Specifications and Requirements

Our specifications can be split up into our hardware specifications and software specifications which are comprised by our software stack. Our hardware specifications at the moment are minimal, a VPS with 1GB memory, 25GB storage, and a single core with 1TB of bandwidth a month. We currently only need a small unified server to run our webapp at the moment which all team members can deploy to and test with, so these minimal specs will suffice.

For the software stack, we will be utilizing Alpine Linux for the server OS, a very light distribution of Linux suited to our small VPS. For the backend portion of our stack, we intend to use MariaDB for the database, Go's Revel for the backend framework, nginx for the webserver, and OpenStreetMap Nominatim API for the location services. MariaDB was chosen as it is essentially an open source version of MySQL, which most

of our group is familiar with. Go's Revel was chosen as Go is an easy language to learn for the portion of our group that does not know it and Revel itself was chosen for it's "full featured" web framework in Go. Nginx was selected simply for being standard fare, ubiquitous, and performant. Finally, the Nomination API was chosen due to it being simple, high quality, and widely used. For our frontend stack, we're just using React, since it's what our frontend developer knows and is very ubiquitous.

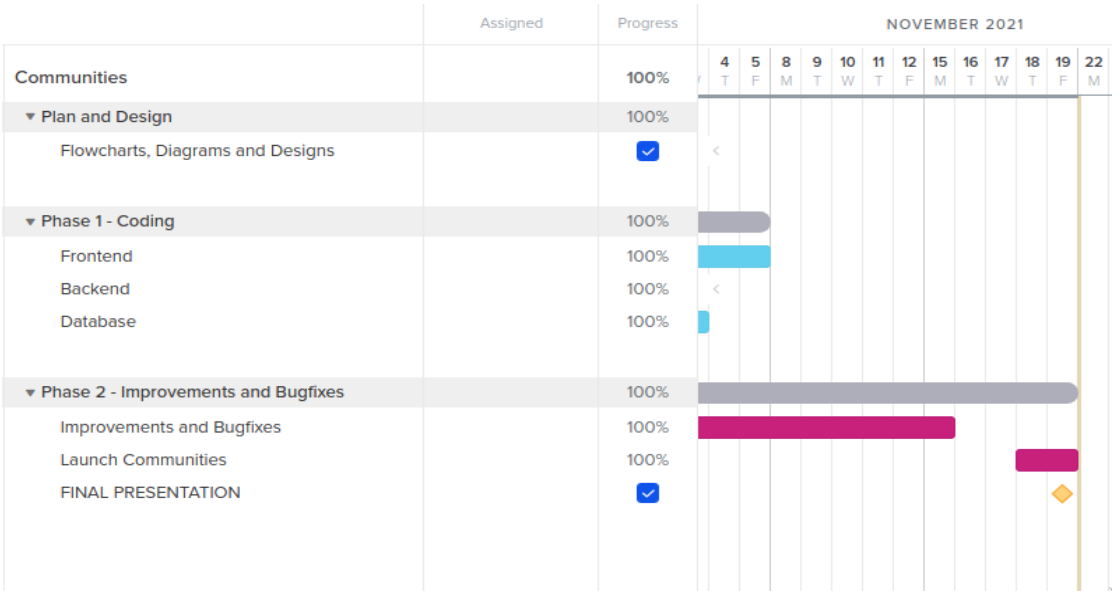
Standards

In so far as standards, we have three main points that we aimed to hit. While our app may be relatively mundane, we will be dealing with potentially very sensitive data given that our program will be using client locational data in order to provide appropriate features. First, we will be taking a look at the International Organization for Standardization or the ISO IEC 20000-1 more specifically concerning our information protection. Currently, our plan is to protect and encrypt our information by making a hash to make the information not readily available. Should this method prove to be not secure enough and we need more, we will look into heavier encryption.

Secondly, our project is concerned with the General Data Protection Regulation or GDPR. In order to follow this, we took precautions in a few ways. The first of which is that we will be asking for client permission before we take any data. Once we have permission, we will want to use it to give them an appropriate. In order to do this we will be limiting the amount of information that we will be giving to the APIs themselves to prevent any problems on that front.

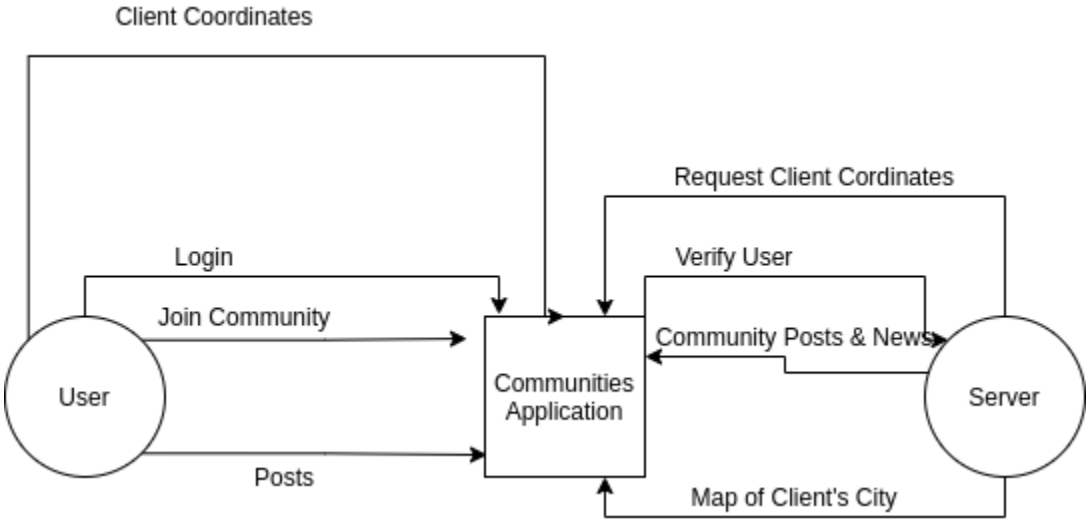
Lastly, this company shall ever be working to adhere to the standard Software Engineering Standards of Ethics. This project in particular is designed to adhere as it is intended to work to the public's interest. Our revenue model allows us to give the app out for free which should further enable local communities to form and thrive. To summarize, our group will be taking precautions and doing preemptive work to keep our project ethical and up to standards.

Project Timeline



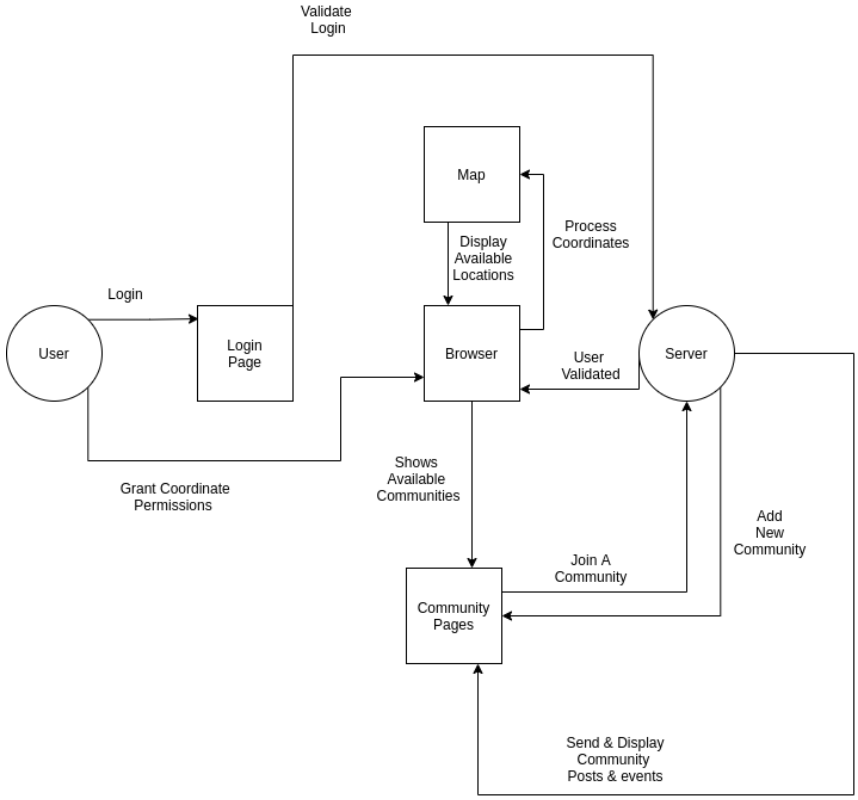
We've finished our initial Phase two of the development process, and our app Communities is now launched. All core functionalities and improvements that we initially envisioned for the project are now fully completed and implemented. Software development remains a continuous process, and we'll continue to add the necessary updates for bugfixes and new functionality to our products in the future.

Level 1 Diagram



Our level 1 diagram consists of the user, the application, and the server. The user will be able to log in, join communities, make posts, and send his/her coordinates. The server will verify the user, and send the maps, communities & news to the application.

Level 2 Diagram

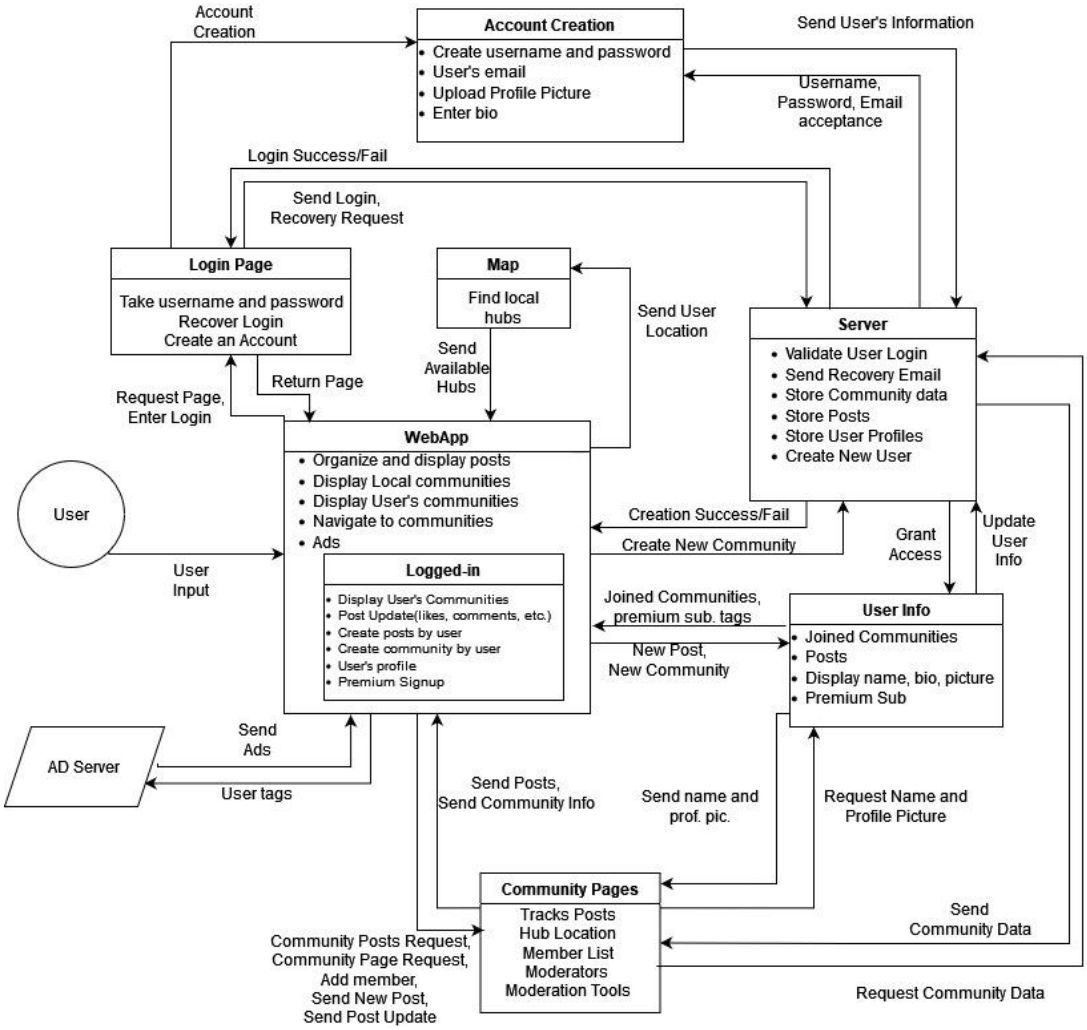


Our level 2 breaks down the application into the login page, browser, map, and community pages. The login page takes in the username and password of the user, which is then validated by the server. After the validation, the user is taken to the browser, which is the webapp for navigating the application consisting of the community pages and the map. The map will process the user's geolocation coordinates and display the user's area. The community pages will display the posts, news, and events from the communities to which the user is subscribed.

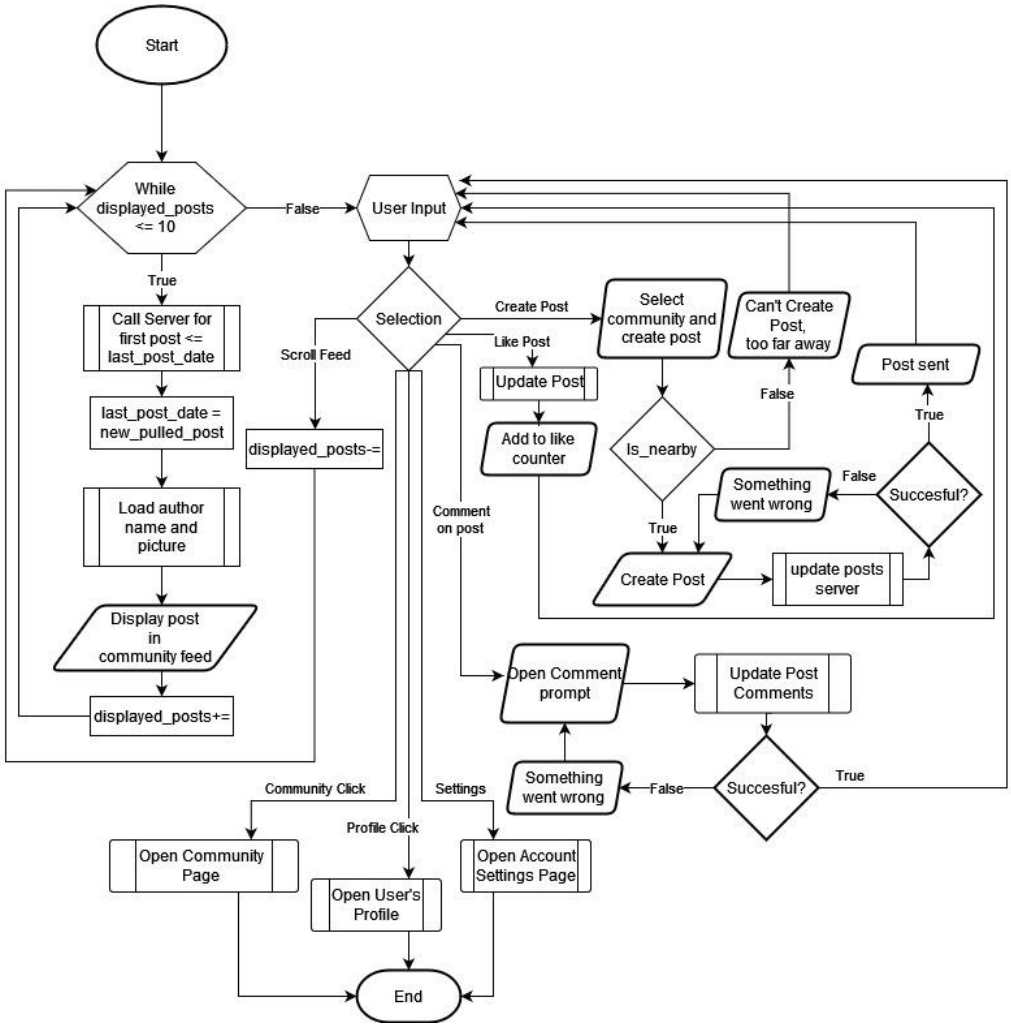
Level 3 Diagram and System Overview

As shown by the level 3 diagram below, each individual process is designed to work with each other. With this design, we can prevent one part from breaking the others in the event of an error. This strategy is helpful for full-scale deployment scenarios and narrowing down bug locations. As you can see, the core functionality of our app allows for user accounts, local communities, and community posts to be created and displayed to the appropriate people.

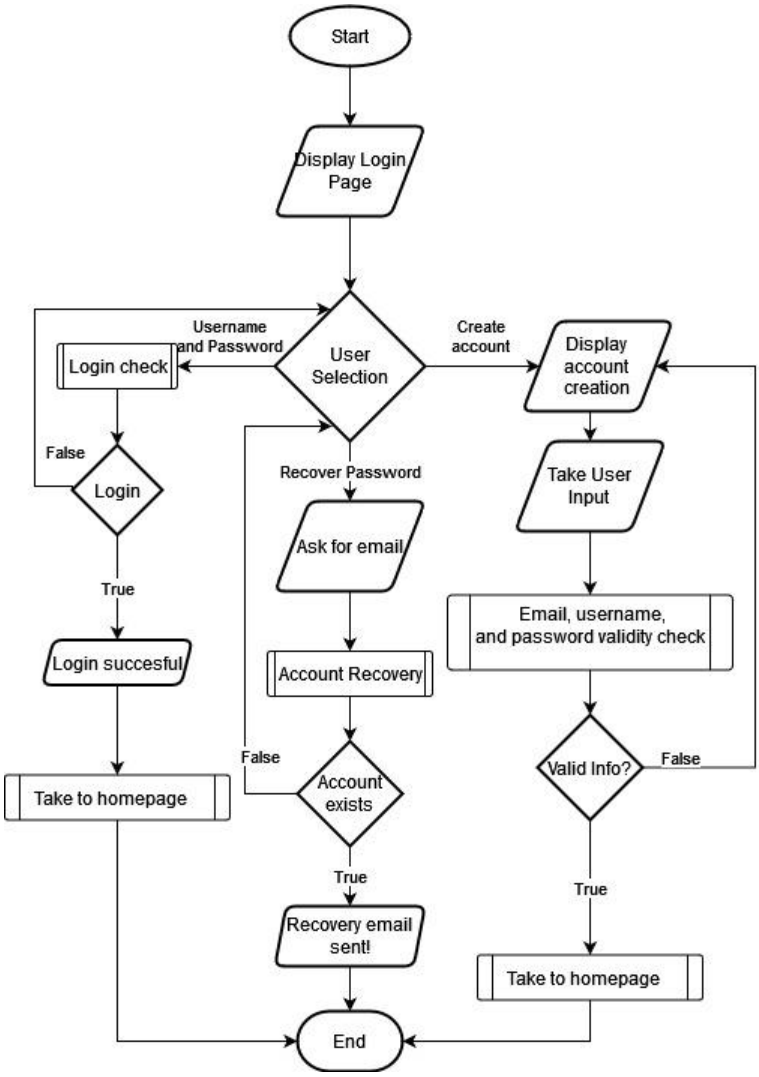
Each of these features piggybacks off one another, so for instance if a user must have an account to create a post, and any given post needs to have an attached community with it, and the users must be within said community's surrounding area in order to post the aforementioned posts.



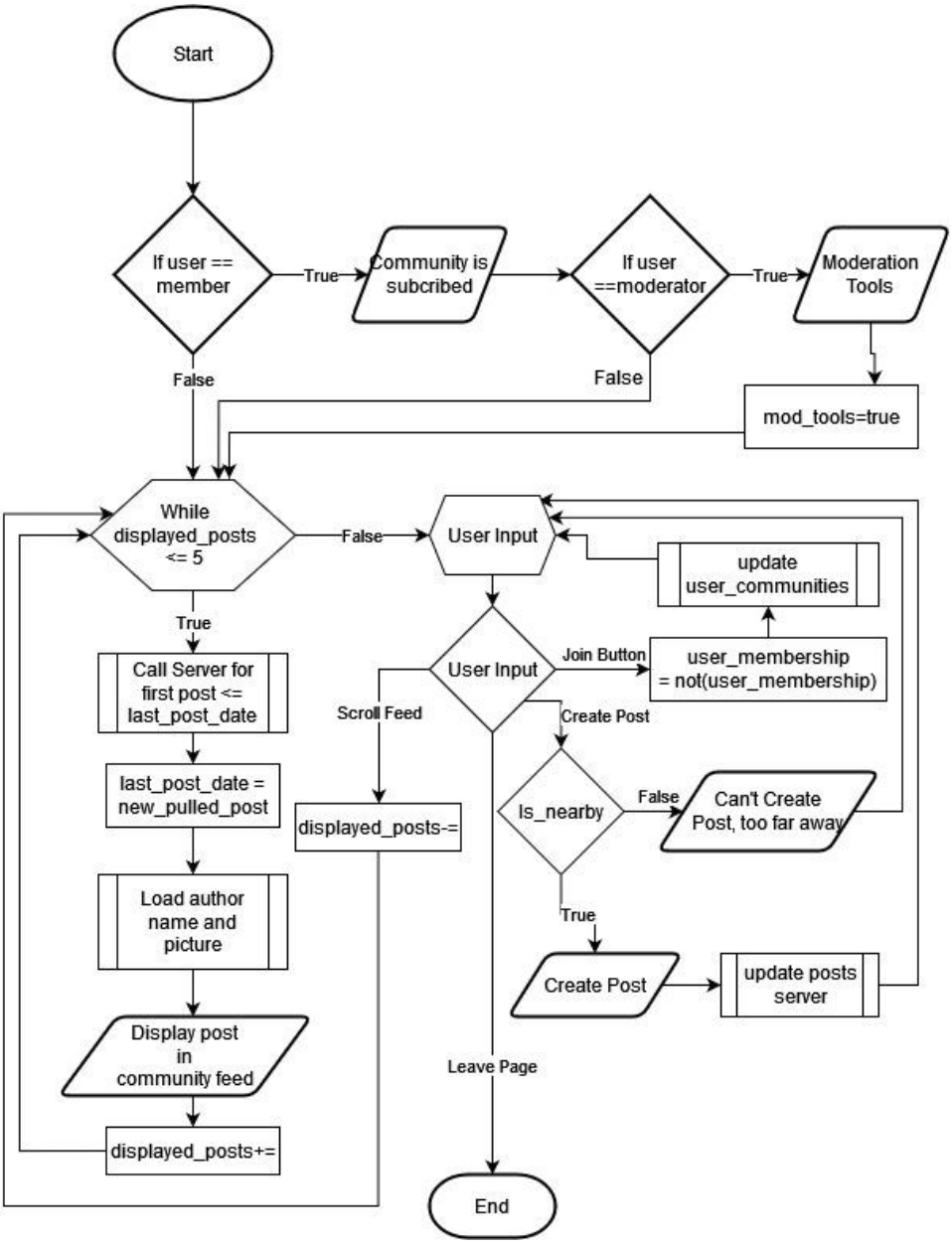
Function Flowcharts



This flowchart shows the home page's sequence of events. The page will display posts, dependent on the user's subscribed communities and location before giving the user their choice of actions to take, be that creating a new post, visiting a community or loading a new set of posts.



This flowchart shows the login process. The process should allow the user to create an account, check for unique email and username, test to see if a password is valid, login, or recover their password. Once any of these actions are complete, the corresponding page will load (either the homepage or login page once again.)



This flowchart demonstrates the functionality of the Community pages. Similar to the homepage, this will display posts specific to that community, from which the user can join the community, create a post, or load more posts, or if the user is a moderator they can also use their special tools to moderator the community.

Technical Design Aspects

As this is a social media-focused app, the most common way users will be interacting with the app is through posts and navigation. As such, we need to ensure

these core functionalities are well-tested and designed. We've designed an interface that would allow the posts being shown to users to be loaded gradually, rather than all at once, to increase performance; we've designed the post creation functions to prevent any bad inputs being sent to the server; and we've assured that the user's options for navigation won't break or lead to a dead end.

Significant Code Snippets and Explanation

Back End Blocks

```
//By default, Index is the first page that loads in Revel
//We are using this to open up our database and make queries.
func (c App) Index() revel.Result {
    //Error that will display if the database connection fails
    var err error

    //Opening the connection to the database
    //This assumes that the database username and password is root
    db, err = sql.Open("mysql", "root:root@tcp(127.0.0.1:3306)/serverstorage")

    //If database fails to connect, display the error mentioning that the database failed to connect
    if err != nil {
        panic(err.Error())
        c.Flash.Error("Database failed to load")
    }

    //Load the communities nearby
    LoadAllCommunities()
    LoadAllPosts()

    //Ping the database in order to ensure that it is connected
    db.Ping()

    //After connecting to the database, redirect to the Login page
    return c.Redirect(App.Login)
}
```

This block covers our Revel implementation which allows us to connect to the database and display the relevant data to our front-end. It first opens our database and, if successful, it grabs the relevant communities and posts and then displays them on our front-end. After it connects to the database, it redirects the user to the login page.

```
//Handles Account Creation
//Called whenever a "CreateAccount" form is submitted
func (c App) CreateAccount(NewUserName string, NewPassword string, NewEmail string, NewPasswordConfirmation string) revel.Result{
    //If passwords do not match, redirect to the Account Creation page
    if(NewPassword != NewPasswordConfirmation){
        c.Flash.Error("Passwords do not match.")
        return c.Redirect(App.AccountCreation)
    }else if(DBCreateAccount(NewUserName, NewPassword, NewEmail, CurrentSess)){
        // If the creation of the account is successful, redirect to the login page.
        c.Flash.Success("Account Created! You may login now. ")
        return c.Redirect(App.Login)
    }
    //If an error occured when creating the account, return to the account creation page.
    c.Flash.Error("Error occured when creating the account, email or username already exists.")
    // defer db.Close()
    return c.Redirect(App.AccountCreation)
}
```

This block handles account creation. It validates the entered password and returns a success message if it successfully communicates with our database and stores the new login. It then checks to see if the username is already in use and redirects back to the account creation with a success message if successful.

```go
//Creates the map for the home page
func createMap(lat float64, lng float64) {
    ctx := sm.NewContext() //Creates a new map 'context'
    ctx.SetSize(1080, 1080) //Sets the size of the map in pixels(?)

    //This should change depending on the city, for now, default to 11.
    ctx.SetZoom(11)




    ctx.AddObject(
        sm.NewCircle( //Draw the circle on the map
            s2.LatLngFromDegrees(lat, lng), //Position by latitude / longitude
            color.RGBA{17, 104, 151, 0xff}, //Outline color
            color.RGBA{150, 212, 231, 1.0}, //Fill color
            20000.0, //Radius
            1, //Weight
        ),
    )

    ctx.AddObject(
        sm.NewMarker( //Draw a marker on the map
            s2.LatLngFromDegrees(lat, lng), //Position by latitude / longitude
            color.RGBA{0xff, 0, 0, 0xff}, // Color of marker
            16.0, //Size of the marker
        ),
    )


    img, err := ctx.Render() //Converts the map data to an image on OSM
    if err != nil {
        panic(err)
    }

    if err := gg.SavePNG("public/img/my-map.png", img); err != nil { //Downloads the image from Open Street Map
        panic(err)
    }
}
```

This block handles the map implementation. Currently, it creates an Open Street Map context at a given latitude and longitude. It then sets the size, zoom, and relevant markers. For the markers, it adds a circle to the map that shows the radius of the city the user is currently in. It also adds a marker in the center of that city to allow the user to differentiate between the different cities on screen. Finally, it renders the map into a temporary image file for ease of use in our front end.

```
//Home Page
func (c App) Home(LoginUserName string) revel.Result{
    //If an attempt is made to access the page without being logged in, remain in Login page
    if(!LoggedIn){
        return c.Redirect(App.Login);
    }
    LoginUserName = ActiveUser

    LatValue, Laerr := strconv.ParseFloat(lat,64)
    if Laerr != nil{
        panic(Laerr.Error())
    }
    LongValue, Loerr := strconv.ParseFloat(lng,64)
    if Loerr != nil{
        panic(Loerr.Error())
    }

    //Create the map for the user to explore
    createMap(LatValue, LongValue)
    //Load the communities nearby
    LoadAllCommunities()
    LoadAllPosts()

    //TODO: Render user communities, latest posts, and communities on the map
    return c.Render(LoginUserName)
}
```

This block handles the home page. It is responsible for loading the map, the posts, and communities within the page. The createMap will not generate a map if the coordinates supplied are NULL.

```
//For each entry in the event table
//Grab all of the details in the event table
//Then spit HTML into the template in order to render the post entry
for allEvents.Next(){
    var EventName string
    var Date string
    var Time string
    var Location string
    var Details string
    readerr := allEvents.Scan(&EventName, &Date, &Time, &Location, &Details)
    if readerr != nil{
        panic(readerr.Error())
    }

    htmlToRender := fmt.Sprintf(`
    <div class = "eventWindow">
        <b> %s </b><br/>
        <i>Date : </i>
        %s
        <br/>
        <i>Time : </i>
        %s
        <br/>
        <i>Location : </i>
        %s
        <br/>
        <br/>
        %s
    </div>
    `, EventName, Date, Time, Location, Details)
    _, htmlRenderErr := EventToHtml.WriteString(htmlToRender)

    if htmlRenderErr != nil{
        panic(htmlRenderErr.Error())
    }
}
```

This block handles the rendering of events within the communities page. For every event that is within our database, an HTML element containing information regarding the event is created.

Front End Blocks

```
<!--
    Defining setups for inputs, buttons, containers
    and forms.
-->
<style>
    .UsernameAndPassword{
        text-align: left;
        height: 500 px;
        padding-bottom: 10 px;
    }
    .userCommunities{
        text-align: left;
        height: 300 px;
        padding-bottom: 10 px;
    }
    .userPosts{
        text-align: left;
        height: 200 px;
        padding-bottom: 10 px;
    }
    input[type=text], input[type=password] {
          width: 50%;
          margin: 8px 0;
          padding: 12px 20px;
          display: inline-block;
          border: 2px solid rgb(0, 0, 0);
          box-sizing: border-box;
    }
    .container {
          padding: 25px;
          border: 3px solid #f1f1f1;
      }
    button {
         background-color: #000000;
         width: 15%;
          color: rgb(184, 182, 179);
          padding: 15px;
          margin: 10px 0px;
          border: none;
          cursor: pointer;
    }
    button:hover {
          opacity: 0.7;
    }
</style>
```

This block is responsible for the styling of our HTML pages. It configures the appearances of elements within the page. This can be seen throughout all of our HTML files.

```
<!--Header Description for the Community that the user is looking at-->
<center><h1> {{.CurrentCommunity}} </h1>{{.CurrentCommunityDescription}}</center>

<!--Go Back button directing to home-->
<form action = "/home" method = "GET">
        <button type="submit" >Go Back</button>
</form>

<!--Container for both componets of the page-->
<div class = "flex-container">
 <!--Left Container for the Posts of the community-->
 <div class = "left-hand-side">
    <!--Header for the Post Window-->
    <div class = "windowHeader">
        <i><b>Latest Posts</b></i>
        <a href = "/NewPost" style="float: right;"><button>Make a new Post</button></a>
    </div>
    <div class = "communityPosts">
     <!--Inclusion of the template that holds all of the rendered posts-->
     {{template "CommunityPosts.html" .}}
    </div>
 </div>
 <!-- Right Container for the events of the community -->
 <div class = "right-hand-side">
    <!--Header for the Event Window-->
    <div class = "windowHeader">
        <i><b>Upcoming events for {{.CurrentCommunity}}</b></i>
        <a href = "/NewEvent" style="float: right;"><button>Make a new Event</button></a>
    </div>
    <div class = "communityEvents">
        <!--Inclusion of the template that holds all of the rendered events-->
        {{template "CommunityEvents.html" .}}
    </div>
</div>
</div>
```

This page is responsible for the HTML of the Community Page. It defines containers and divs for the Post and Event Windows. The HTML for the posts and events is found within the templates “CommunityPosts.html” and “CommunityEvents.html”

Version Control and Associated Programs

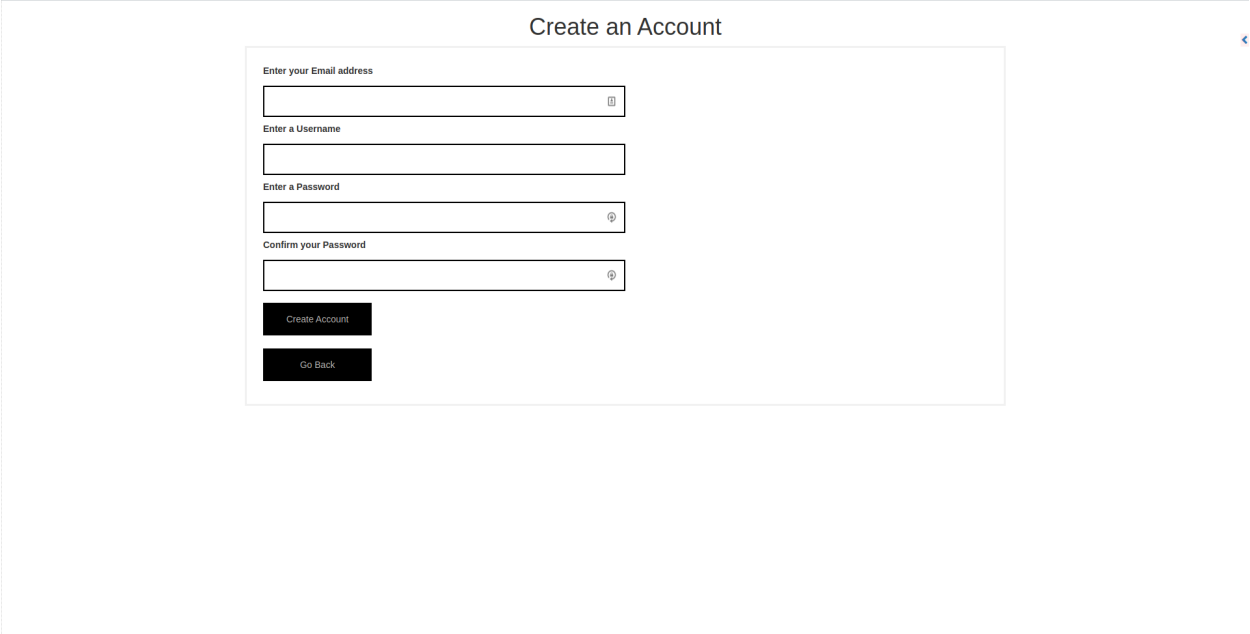
The version control that we utilized was Github. We used Github to store various versions of our applications, allowing us to test out new functionality and features before release. We had our main branch, a branch for server usage, a branch for live map implementation, and a branch for conducting standard bugfixes and enhancements. We also used Github as a method of tracking all issues and enhancements that needed to be done for our application. In order to conduct meetings and discussions, we used Discord.

System Implementation and Testing

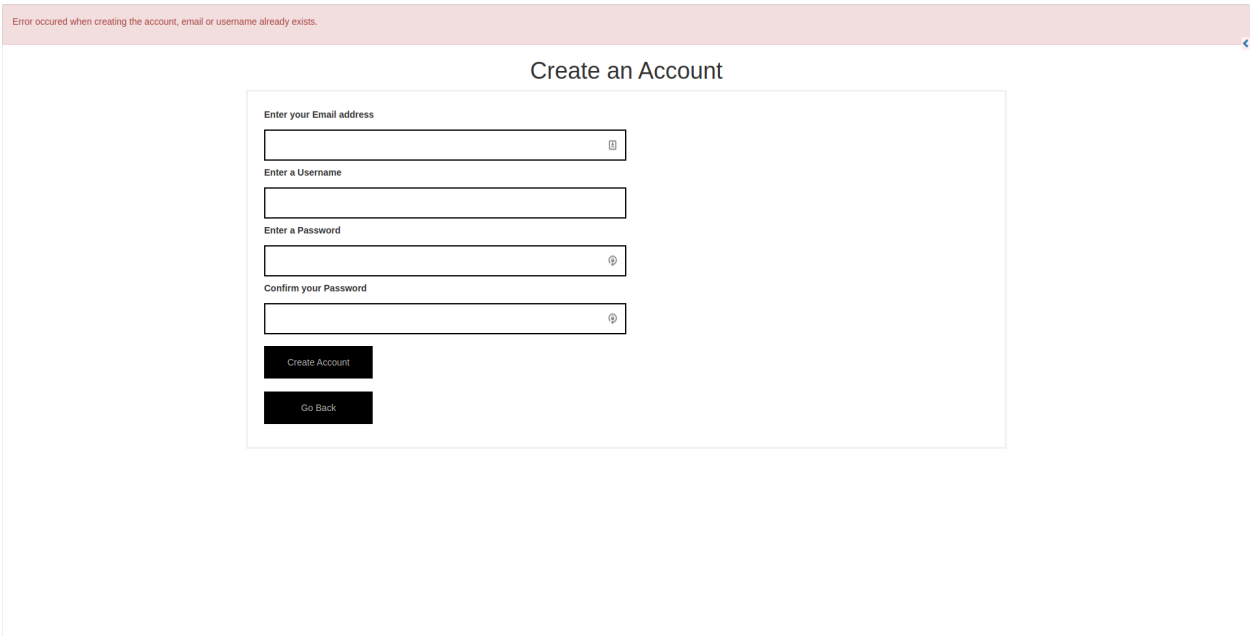
The system was implemented via Revel, our chosen web framework. Revel is a full stack framework that utilizes Go for the backend, and HTML for the frontend. We also utilized MariaDB for the creation of our database of users, communities, and posts.

Whenever we want to run a test on our application, we first spin up the server that is holding our database and then run our application via the terminal command `revel run Communities`. The command builds our website and allows us to test any features that we wanted to run.

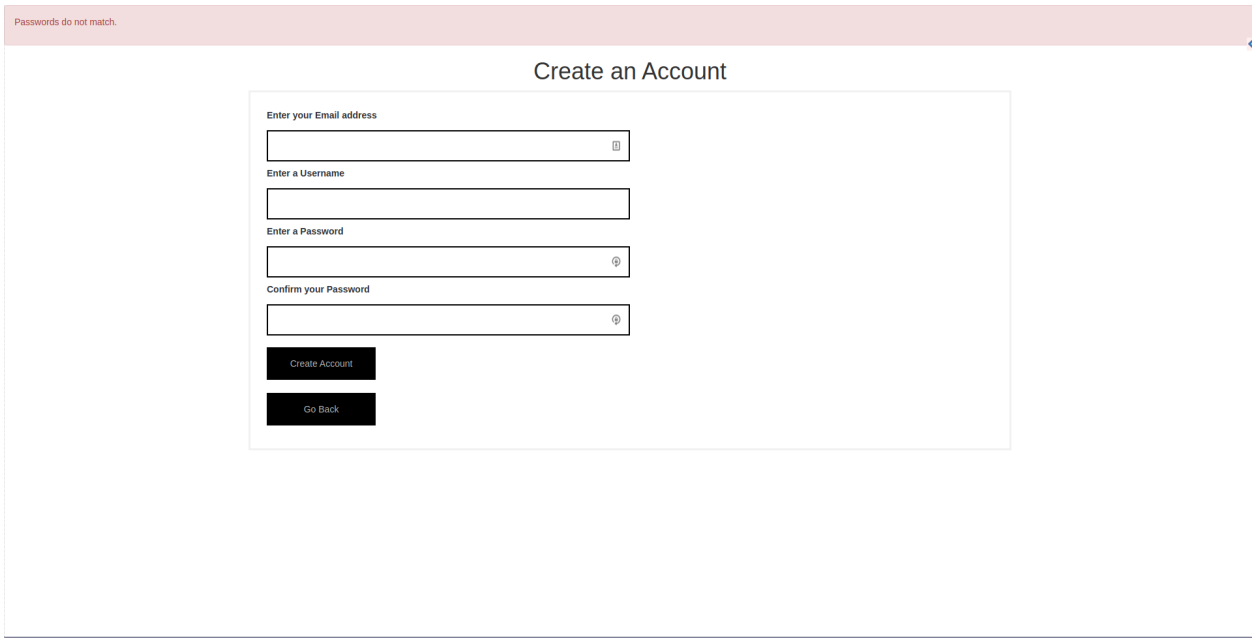
Evidence of Functionalities of the Working Project



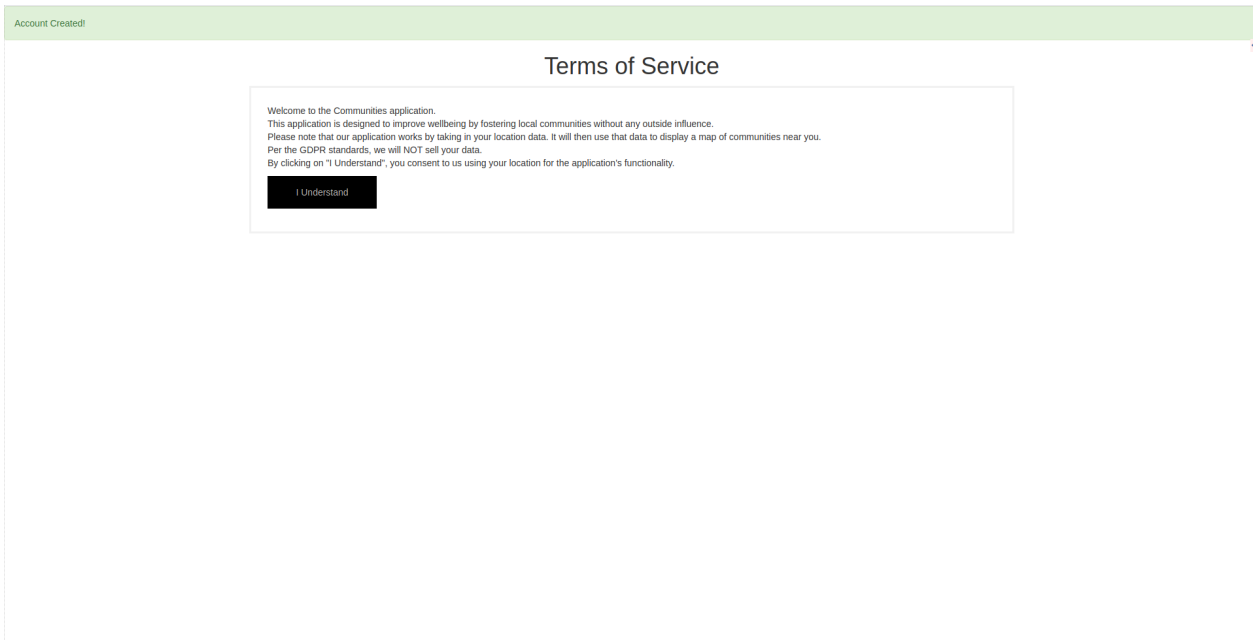
This is our Account Creation page. Users can create an account with an email address, username, and password. The password is prompted twice in order to prevent accidental inputs in it.



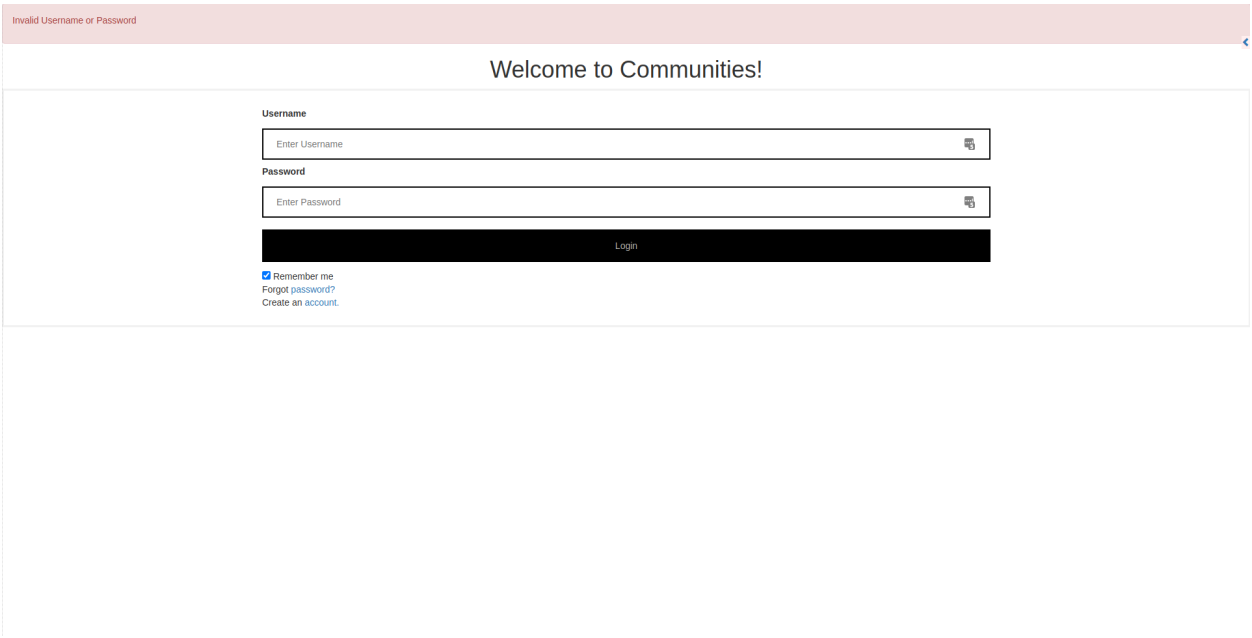
If a user attempts to create an account with an email or username that is used by another user, the error message “Error occurred when creating the account: Username or email already exists” displays.



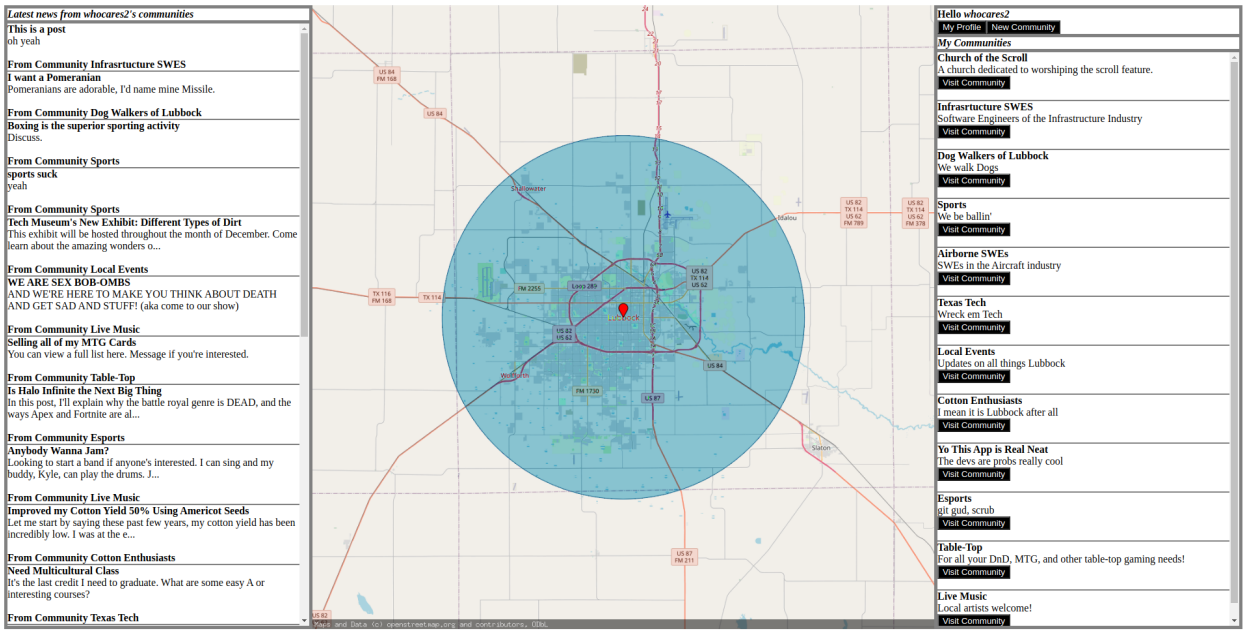
Similarly, if a user does not input the same password on both the “Enter a password” and “Confirm your Password” fields, the error message “Passwords do not match” displays.



When a user has successfully created an account, he/she is directed to the Terms of Service page. When “I Understand” is clicked, the user will be able to login.

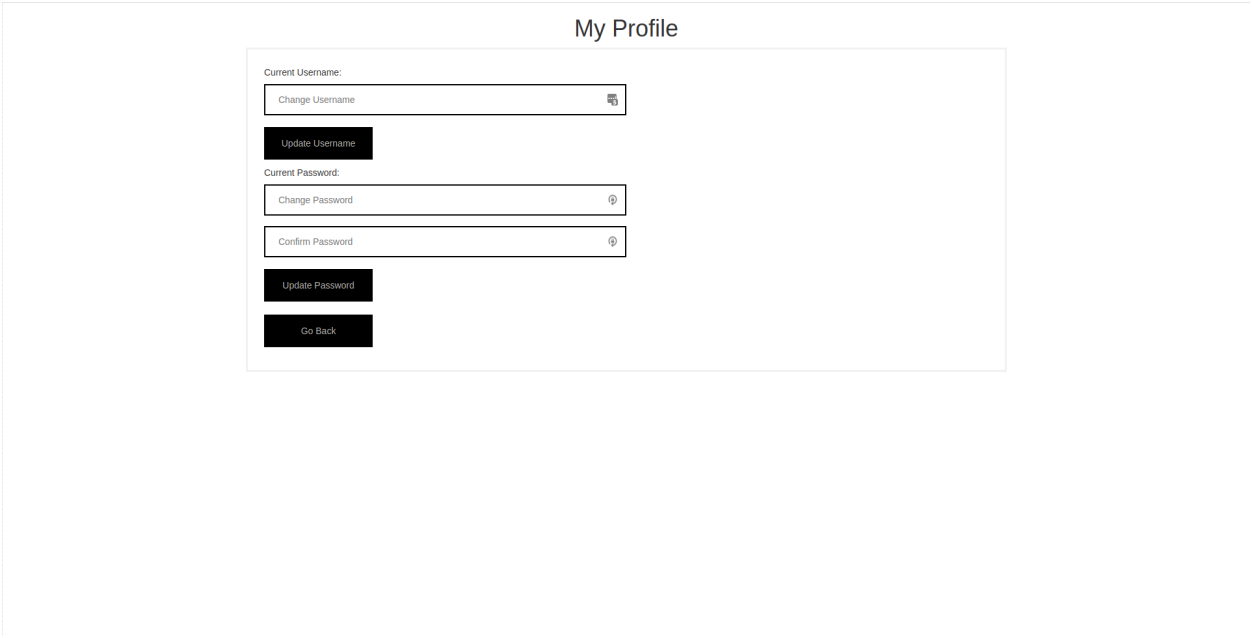


When the user inputs a username or password that does not line up with their credentials, the message “Invalid Username or Password” displays.

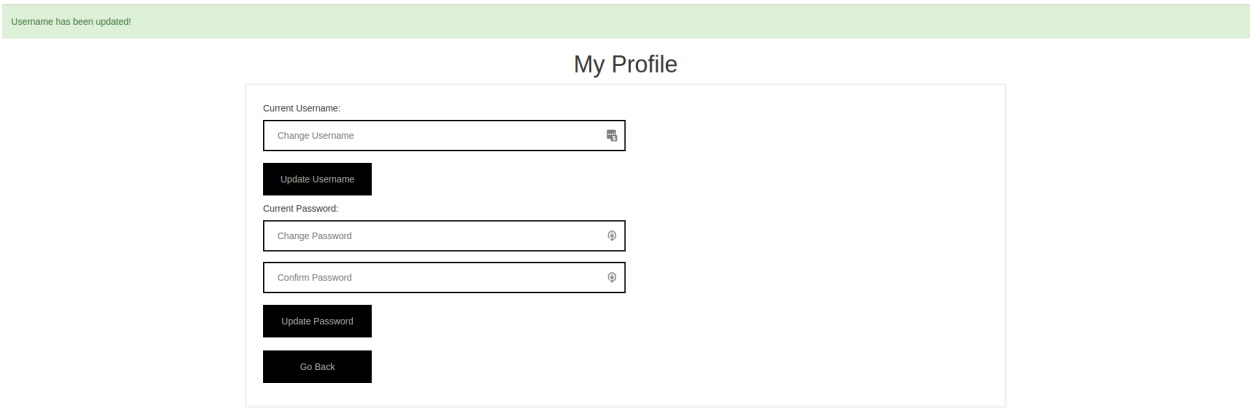


Once the user logs in, he/she is taken to the home page. The latest posts are displayed on the left, the map is displayed in the center, and the profile window/Communities page displayed on the right. Whenever the Post or Scrolling Window overloads with posts, a scrolling bar will appear on the right. Currently, the map only shows the area of the user.

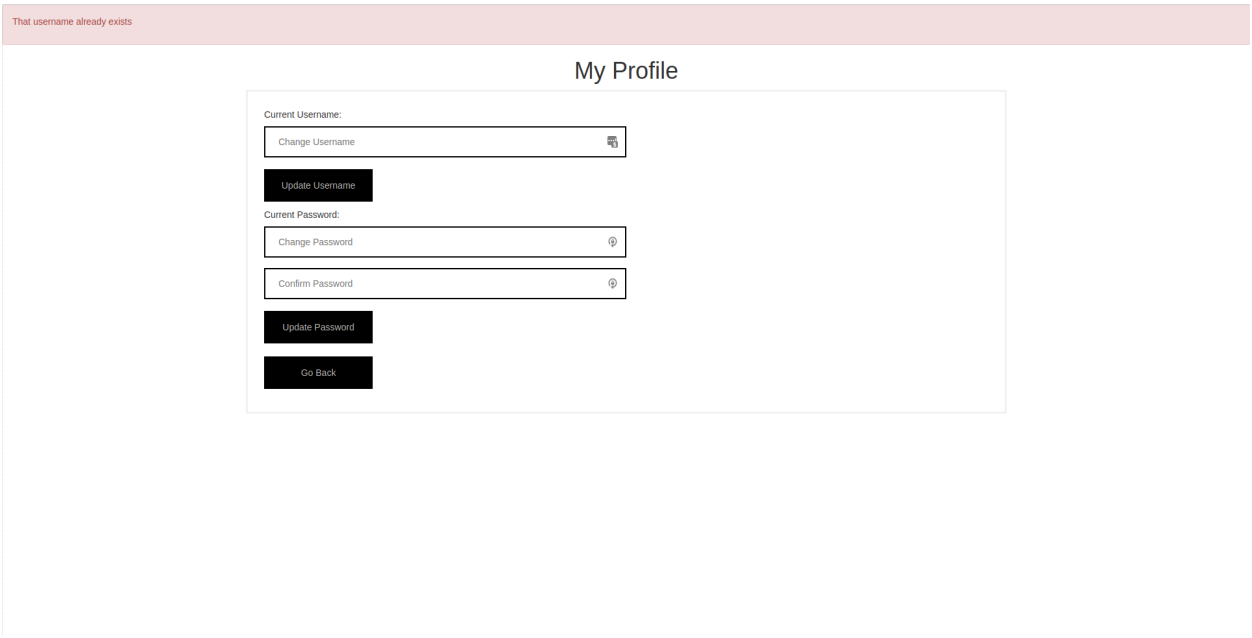
On the next release, the map will show the locations of communities that the user can subscribe to.



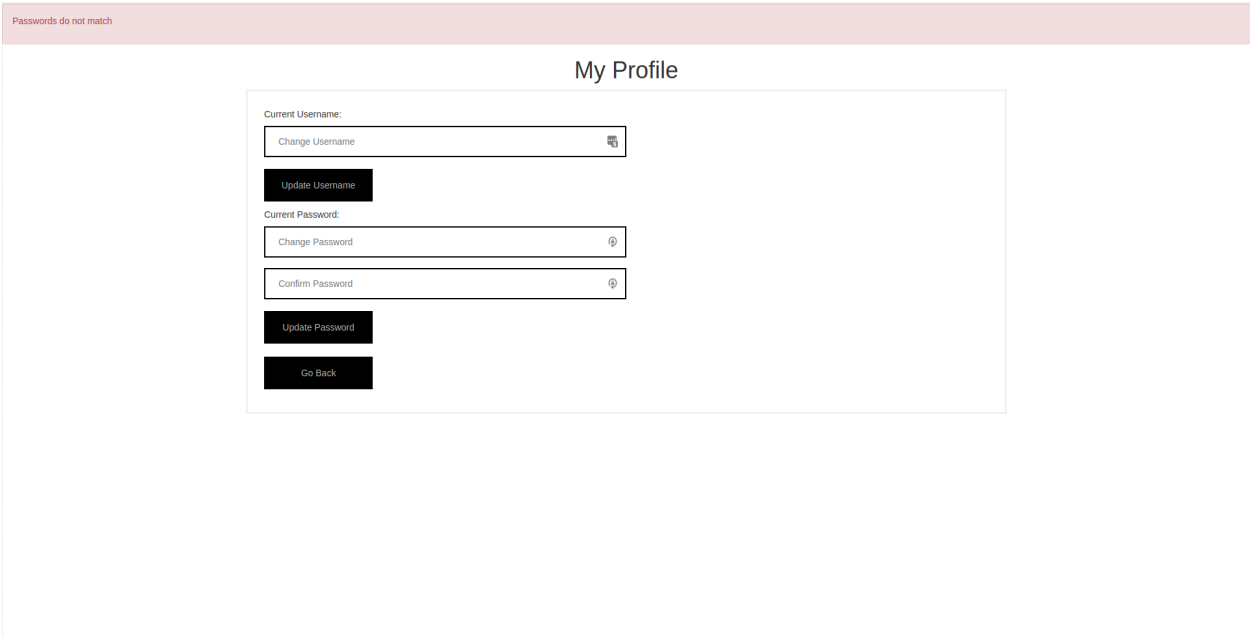
When the user wants to change details of his/her profile, he/she can update the profile details via this page. It is accessed via the “My Profile” button in the profile window of the home page. The user can change the username or password.



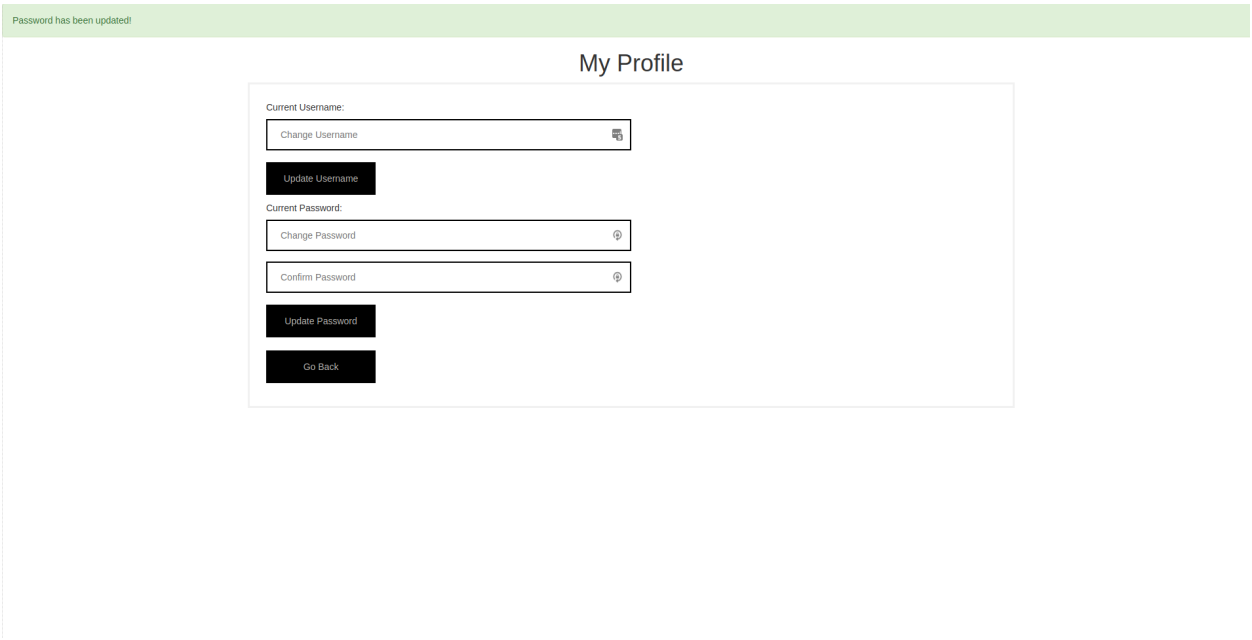
As long as the username is not already taken, the user is free to change his/her name.



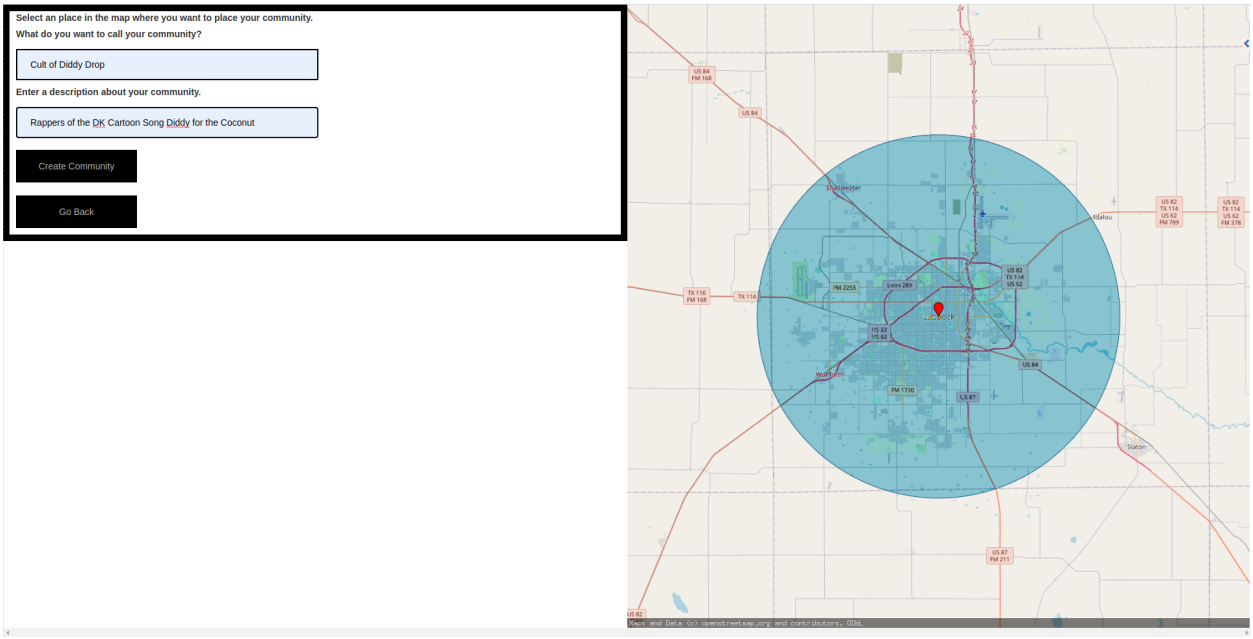
If the user attempts to update his/her username with a name that is already taken, the message “That username already exists” displays.



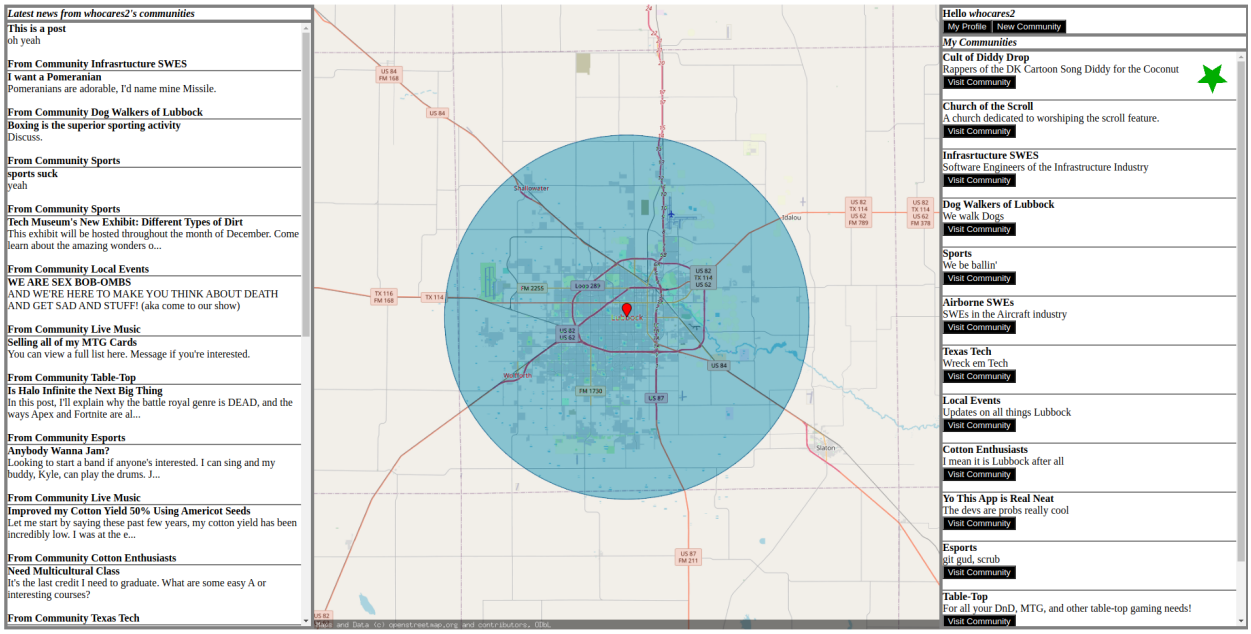
If the user attempts to update his/her password with two different passwords in “change password” and “confirm password‘ fields, the message “Passwords do not match” displays.



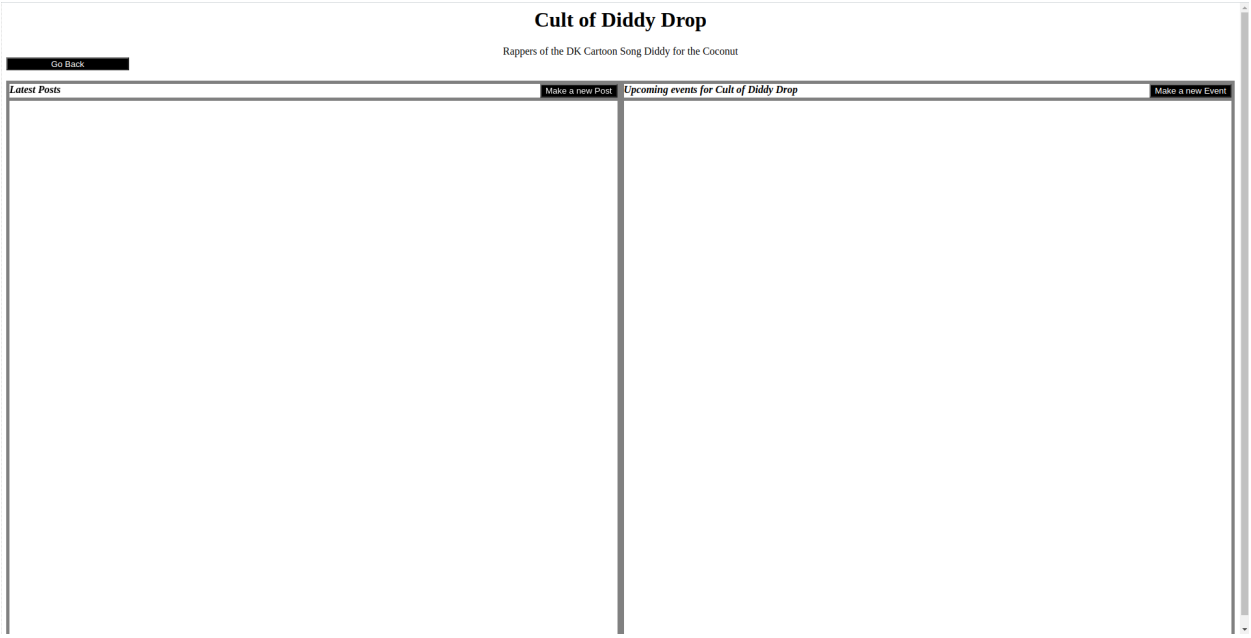
When the user enters a new password and confirms it successfully, the message “Password has been updated!” displays.



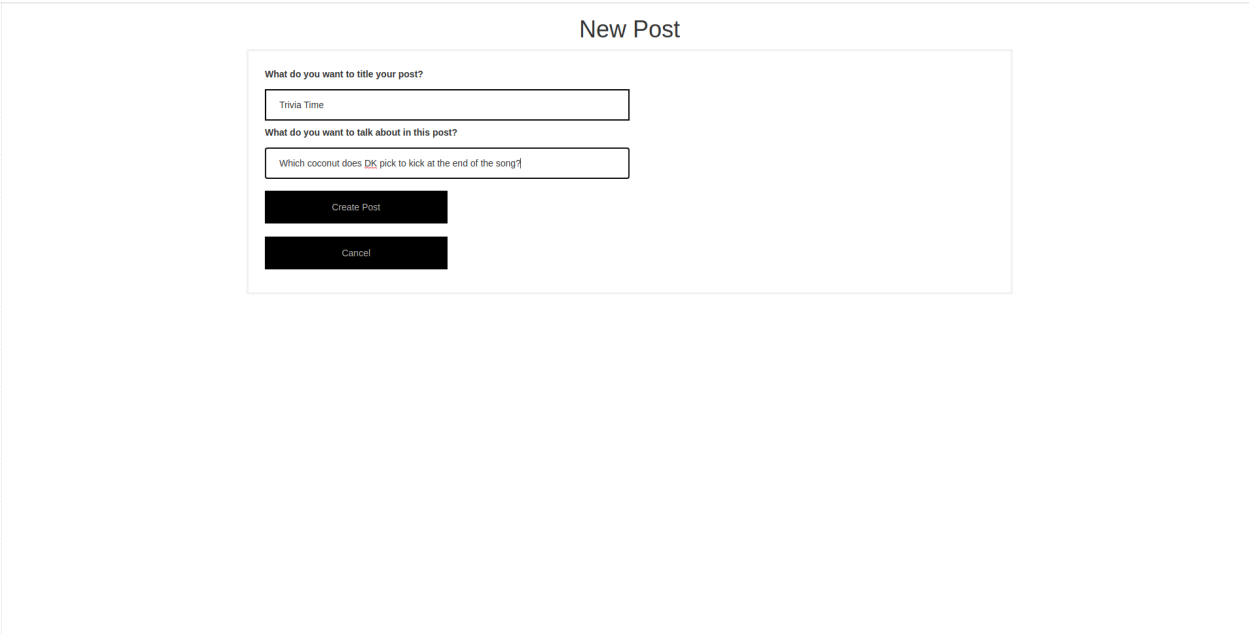
This page is for the creation of a new community. It is accessed via the “Create New Community” button in the profile window. The user enters what he/she wants to call the community and the description of it. Eventually, the user will be able to click on the map in order to place the location of the community.



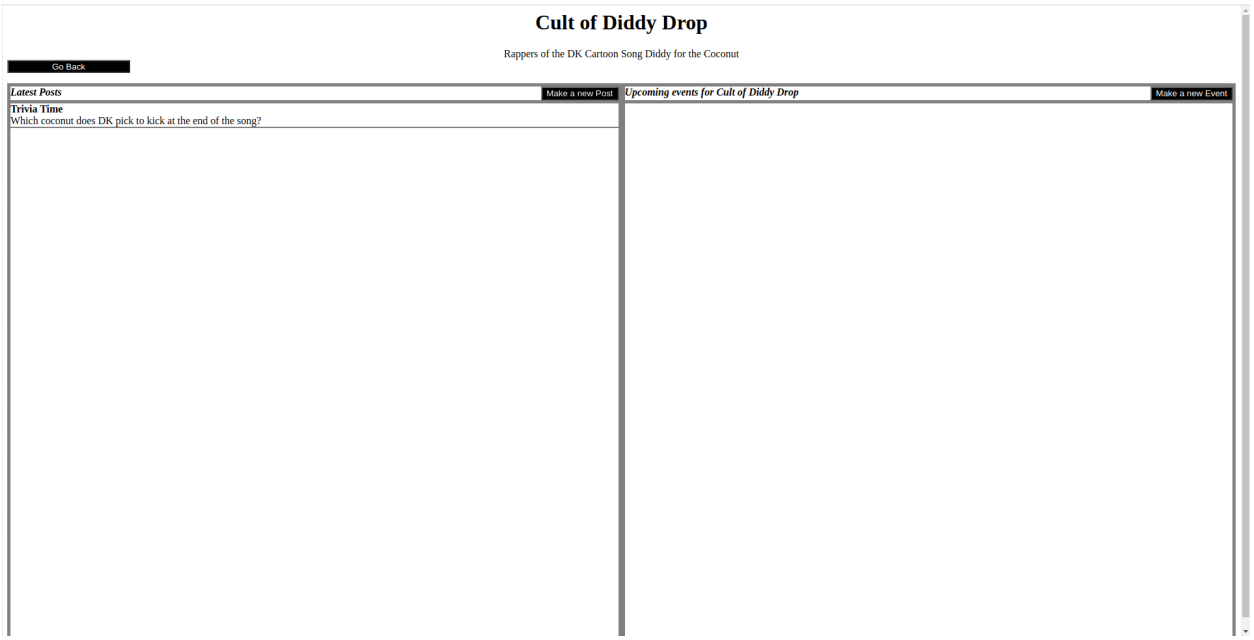
Once the community is created, it will show within the communities window on the right. Eventually, it will also show up on the map. The new community is marked with a green star. In our previous version, we had the new communities appear at the bottom of the page. In this version, we have new communities appear at the top.



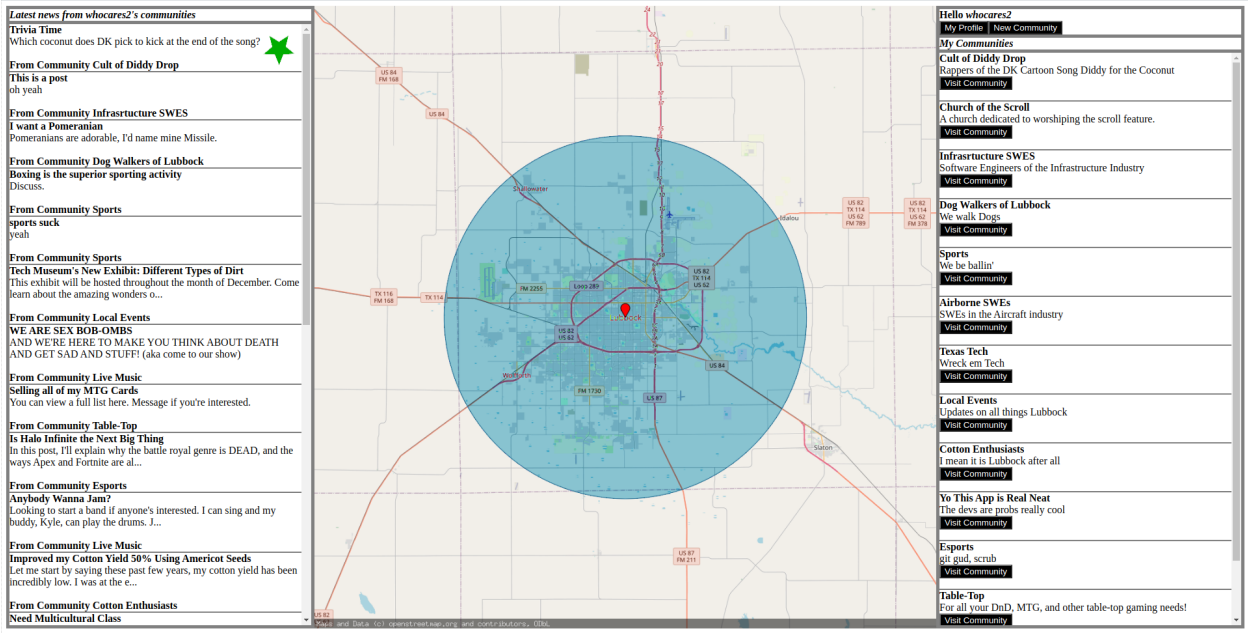
When the user clicks on “Visit Community” for a particular community, he/she will be taken to the community hub page. The hub will show the posts, events, and description of the community that the user visited.



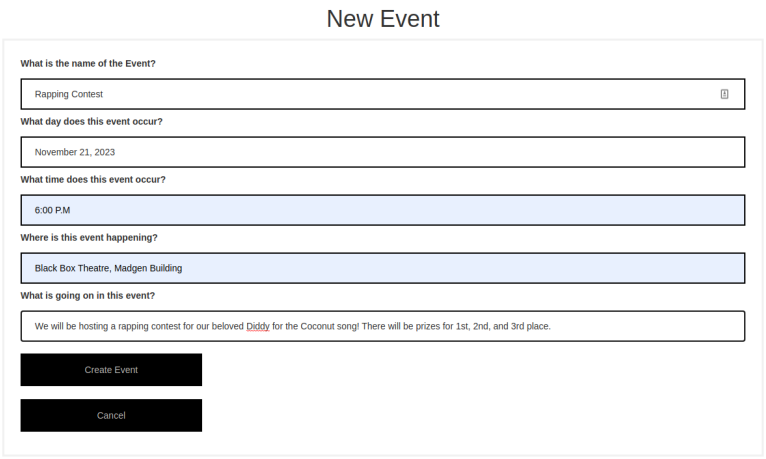
This page is for creating new posts. It is accessed via the “Make a new post” button on the Community hub located in the post window.



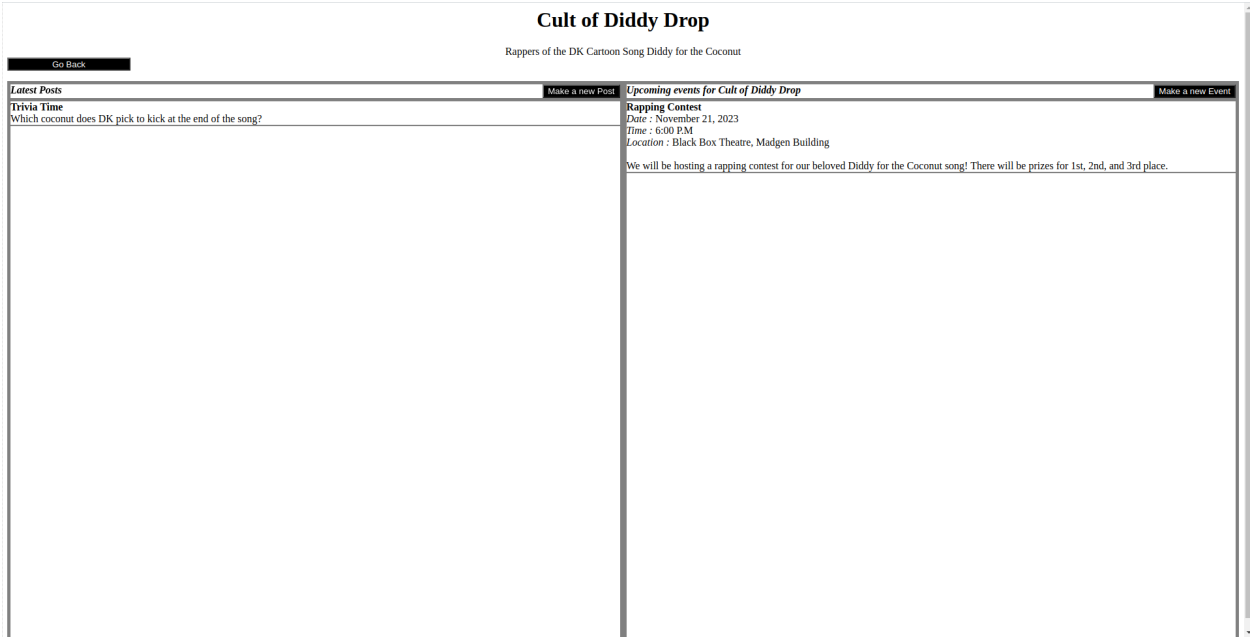
After the post is created, it will appear in the community's post window.



The new post is also displayed in the post window at the home page. The new post is marked with a green star. All post also display which community they came from.



This page is for creating a new event. It is accessed via the “Make a New Event” button on the right of the page.



After the event is created, it appears on the event window to the right.

Conclusion

Our program is designed to help provide a new platform where people can interact with each other on the local level. Taking this a step further, we endeavor to further encourage interaction via the map and through our event planning. Ideally, this app should significantly impact people for the better. Should we continue to work on this app, we would like to add further quality of life features and add further features to maintain our user base.

As a group, we have significantly improved in several important aspects. First, we have improved our ability to work together through meetings and careful planning. As a result, we have learned what needs to be communicated and how to manage our workload in order to complete our tasks on time. Additionally, we have improved our teamwork and synergized much better while working towards the completion of our project.

References

Carlile, Liz. *Development Online: Making the Most of Social Media.* International Institute for Environment and Development, 2011, www.jstor.org/stable/resrep01460. Accessed 13 Sept. 2021.

Nair, Madhu. “Social Media Effects on Communication.” *University of the People*, 7 Jan. 2021, https://www.uopeople.edu/blog/how-social-media-affected-communication/.

Pantalone, Maria. “Social Media and Its Impact on Communication Skills.” *Infinite Growth*, 5 Nov. 2021, https://infinitegrowth.com.au/social-media-and-its-impact-on-communication-skills/.